# Experiences with DAMED-WG Events

Authors: Dan Gunter, Brian Tierney. {dkgunter,bltierney}@lbl.gov

## *Introduction*

The DIDC group at Berkeley Lab has applied the DAMED ideas for event naming to come up with a set of NetLogger messages for some network measurement tools including ping, iperf, netest (from LBNL) and web100 variables, as well as some host measurements including CPU load and free memory. These messages are being stored into a MySQL database. We are developing scripts and web interfaces to pull this data out and analyze it.

Because some of our decisions may have been influenced by our message transport protocol, NetLogger, a brief description of it is necessary. NetLogger messages are timestamped and contain a series of *name* and *value* pairs, called fields. In the monitoring context discussed here, there are several required fields: HOST for the sending host, PROG for the monitored or sending program, NL.EVNT for the event name. If there is a numeric value for this event, by convention it is placed in a field called VAL. A typical event looks like this:

```
DATE=20020926202203.141870 NL.EVNT=web100.SampleRTT HOST=140.173.170.41
PROG=web100_iperf_c PID=30661 DST=140.173.155.14 VAL=70
```

## *Modifications to DAMED-WG Events*

This process has led us to try several modifications to the basic framework proposed in the DAMED "Top 'N' Events" document:

1. Some of the names were modified, particularly the network names
    a. These changes were made to accord with our view of the NM-WG consensus
2. The program generating the events was not made part of the event name, but perhaps it should be.
    a. On the one hand, this is redundant information because every NetLogger event contains a required PROG field.
    b. On the other hand, without the program name in the event name, distinguishing between bandwidth returned by (for example) *iperf*and *netest* requires two-steps and is database/implementation-dependent.
3. Required parameters were added for some events
    a. VAL was added to *all* events, and we feel this represents the spirit of the DAMED work so far that events are very close to single-valued attributes in a larger schema
    b. Some programs report periodic results, then summarize them at the end. We usually want to log both the periodic reports and the summary. In order to do this we use a standard field REPORT which can have value “INTERVAL” or “RUN”
    c. NUM_STREAMS and BUFFER_SIZE was added to active TCP tests
    d. HOP_NUM was added to multi-hop tests (maybe this should be part of the target?)

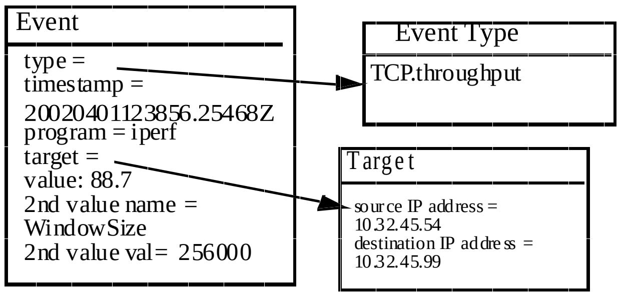
    e. To distinguish between different ways of measuring RTT, we added the standard field METHOD. Arguably, the value of this field should be appended to the event name.
4. The event "target" was specified in a standard way
    a. SRC, DST indicate the source and destination host. In the case where there is only a source, DST is missing or empty
5. Parameters were added to represent aggregation explicitly
    a. AGG_TYPE represents the type of aggregation, e.g. mean, median, deviations, etc.
    b. AGG_IVAL indicates the interval over which the data was aggregated
    c. This was done to map to tests that periodically report averages such as iperf.
    d. We haven't used this much, yet, but feel that some support for aggregation needs to be built-in to the event model rather than invented ad-hoc for each new one.

These modifications can be seen in the spreadsheet of our events, below. In this spreadsheet, the program then original ad-hoc event name is listed first, then the DAMED name we chose, the default type of aggregation and units, a description of the event, and finally the parameters needed to describe the target as well as any "additional" parameters specific to this type of event.

| program | old event name | new event name | aggregation type (AGG_TYPE) | aggregation interval (AGG_IVAL | units (UNITS) | description | target parameters | additional parameters |
|---|---|---|---|---|---|---|---|---|
| Iperf | TCP.WIN.SIZE | *BUFFER_SIZE attribute of bandwith.achievable* | - | - | - | - | - | - |
| | TCP.THROUGHPUT.RUN | bandwidth.achievable.tcp.singleStream | mean | (sec) | Mbits/s | mean throughput for a run, with 1 stream | SRC, DST | REPORT=RUN, BUFFER_SIZE |
| | | bandwidth.achievable.tcp.multiStream | mean | (sec) | | mean throughput for a run, with parallel streams | SRC, DST | REPORT=RUN, NUM_STREAMS, BUFFER_SIZE |
| | TCP.THROUGHPUT | bandwidth.achievable.tcp.singleStream | mean | (sec) | Mbits/s | mean throughput for one reporting interval, with 1 stream | SRC, DST | REPORT=INTERVAL, BUFFER_SIZE |
| | | bandwidth.achievable.tcp.multiStream | mean | (sec) | | mean throughput for one reporting interval, with parallel streams | SRC, DST | REPORT=INTERVAL, NUM_STREAMS, BUFFER_SIZE |
| GridFTP | ?? | bandwidth.achievable.tcp.singleStream | mean | (sec) | | mean throughput for one run, single-stream | SRC, DST | BUFFER_SIZE |
| | | bandwidth.achievable.tcp.multiStream | mean | (sec) | | mean throughput for one run, parallel streams | SRC, DST | NUM_STREAMS, BUFFER_SIZE |
| Ping | ping | delay.roundTrip | mean | (sec) | ms | Average of ping tests | SRC, DST | METHOD=ICMP |
| netest | RTT | delay.roundTrip | avg | (sec) | ms | Round trip delay | SRC, DST | METHOD=(ICMP\|UDP\|TCP) |
| | S.THRU.UDP | bandwidth.achievable.udp.singleStream | max | (sec) | Mbits/s | Maximum UDP bandwidth from this host, with 1 stream | SRC, DST | |
| | M.THRU.UDP | bandwidth.achievable.udp.multiStream | max | (sec) | Mbits/s | Maximum UDP bandwidth from this host, with parallel streams | SRC, DST | NUM_STREAMS |
| | OptWIN | tcp.recommendedBufferSize | N/A | N/A | bytes | recommended max buffer size for TCP | SRC, DST | |
| | S.THRU.TCP | bandwidth.achievable.tcp.singleStream | | | Mbits/s | Predicted achievable throughput for a single stream | SRC, DST | MIN, MAX, AVG |
| | RECOMMEND_MULTI_STREAMS | bandwidth.achievable.tcp.recommendedStreams | N/A | N/A | int | Is the bandwidth from TCP improved by using parallel streams? | SRC, DST | |
| | BW.USED.BY.TCP.TEST | application.bandwidth.used.tcp | N/A | N/A | | TCP bandwidth used by the netest | SRC, DST | |
| pipechar | HopN_THRU | hop.bandwidth.capacity | N/A | N/A | Mbits/s | Estimated capacity for a single hop on a path. | SRC, DST, HOP_SRC, HOP_DST | HOP_NUM=<N> |
| | HopN_CNGST | hop.bandwidth.utilized | N/A | N/A | Mbits/s | Estimated utilized capacity for a single hop. | SRC, DST, HOP_SRC, HOP_DST | HOP_NUM=<N> |
| | path_bottleneck | path.bandwidth.available.bottleneckHop | N/A | N/A | Mbits/s | Available bandwidth of the bottleneck hop in the path | SRC, DST | HOP_NUM=<N> |

## *Placing DAMED Events in an Relational Database*

We wanted to store historical records of NetLogger events in a MySQL database. In order to do this, we devised a simple but general-purpose schema, illustrated below with some specific values filled in:



The SQL statements to create the schema are (does anyone have a tool to make a drawing from this?):

```
--
-- Table structure for table 'EventType'
--

CREATE TABLE EventType (
  evnt_type int(11) NOT NULL default '0',
  event varchar(25) default NULL,
  PRIMARY KEY  (evnt_type)
) TYPE=MyISAM;

--
-- Table structure for table 'Events'
--

CREATE TABLE Events (
  date double(11,9) NOT NULL default '0.000000000',
  program varchar(32) NOT NULL default '',
  val double(11,9) default NULL,
  group_id int(11) NOT NULL default '0',
  target_id int(11) NOT NULL default '0',
  evnt_type int(11) NOT NULL default '0',
  evnt_id int(11) NOT NULL auto_increment,
  PRIMARY KEY  (evnt_id),
  KEY evnt_type (evnt_type),
  KEY date_idx (date),
  KEY prog_idx (program)
) TYPE=MyISAM;

--
-- Table structure for table 'NumVals'
--

CREATE TABLE NumVals (
  evnt_id int(11) NOT NULL default '0',
  name varchar(25) default NULL,
  val double(11,9) default NULL,
  KEY evnt_id_idx (evnt_id)
) TYPE=MyISAM;

--
-- Table structure for table 'StrVals'
--
```

```
CREATE TABLE StrVals (
  evnt_id int(11) NOT NULL default '0',
  name varchar(25) default NULL,
  val varchar(25) default NULL,
  KEY evnt_id_idx (evnt_id)
) TYPE=MyISAM;

--
-- Table structure for table 'Targets'
--

CREATE TABLE Targets (
  target_id int(11) NOT NULL auto_increment,
  src varchar(16) default NULL,
  dst varchar(16) default NULL,
  PRIMARY KEY  (target_id)
) TYPE=MyISAM;
```

## *Open Issues*

Several open issues have been noted above.